CAAP Quarterly Report

Date of Report: March 29, 2020

Prepared for: *U.S. DOT Pipeline and Hazardous Materials Safety Administration*

Contract Number: 693JK31950002CAAP

Project Title: AI-enabled Interactive Threats Detection using a Multi-camera Stereo Vision System

Prepared by: Arizona State University

Contact Information:
Dr. Yongming Liu (PI), Email: Yongming.Liu@asu.edu
Dr. Yang Yu (Co-PI), Email: yangyu18@asu.edu

For quarterly period ending: March 29, 2020

Business and Activity Section

(a) Contract Activity

Impact of COVID-19

Following the suggestions from PHMSA, we are reporting impact/predicted impact of the COVID_19 to the executions of the proposed study.

1. Any major slowdown of project scope execution due to impacts to the contract recipient or from your subcontractors.
   Answers: Due to the COVID-19, ASU changed the teaching to be online teaching for the entire spring semester and all faculties/students are advised to work from home. This has minimal impact on the algorithm development and numerical studies in the proposed study. This does have impact for the experimental testing and prototype development as lab is not accessible and many staff members are not on site. We implemented risk mitigation plans for this situation: a) use openly available database to demonstrate the methodology; b) use simple demonstration testing of images acquired in apartment for illustration and algorithm validation; c) re-schedule the numerical and experimental task progress; 4) experimental testing is planned to be resumed once the COVID-19 pandemic is over. We did not expect this will have significant impact to the proposed study as this is a three-year project and we have sufficient time to catchup the prototype delay in the future 10 quarters.

2. Any business judgement call to cease research operations due to impacts to the contract recipient or from your subcontractors. Meaning you will no longer be executing any scope on your project for the foreseeable future.
   Answers: None at this stage. Please see detailed risk mitigation plan in the above description.

Discussion about contract modifications or proposed modifications:
None

Discussion about materials purchased:

1. USB cable
2. Battery

(b) Status Update of Past Quarter Activities

In this quarter, the research team works on Task 1.1 and Task 2.1 to develop algorithms for estimating distance using stereo vision and unsupervised image segmentation using disparity map and color images. Experiments were conducted to verify the effectiveness and accuracy of the developed algorithms.

Student Training Activities

- Kailing Liu (MS student) works on stereo vision algorithm development for pipeline imaging, distance estimation, and preliminary demonstration to test the effectiveness and accuracy of the developed algorithm. (Task 1)
- Rahul Rathnakumar (PhD student) works on developing an unsupervised algorithm for image segmentation as a method for preprocessing the image for pipeline defect detection. (Task 2)

(c) Cost Share Activity

All cost share requirements have been satisfied in the past quarter and detailed financial report will be submitted by ASU financial department.

(d) Detailed Description of Work Performed

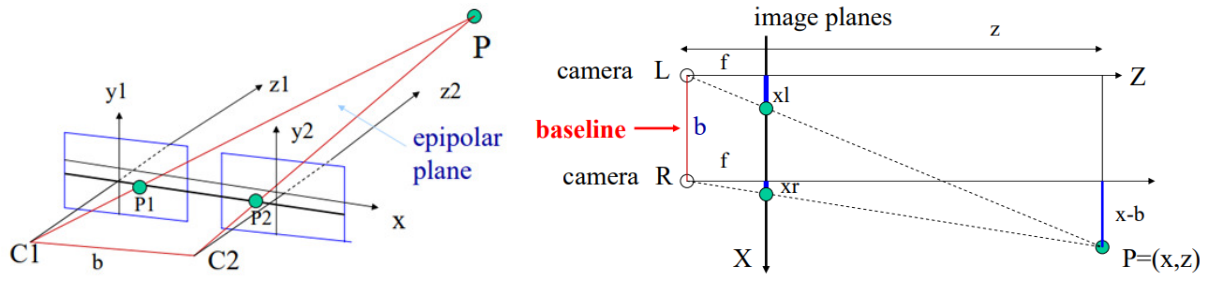
1. Background and Objectives in the 2nd Quarter

Pipeline anomalies such as fatigue cracks, stress corrosion cracking, corrosion pits, and seam weld defects are major threats to the integrity of pipeline systems. The detection and characterization of these pipeline anomalies are critical for the safe operation of pipeline infrastructure, which is the objective of this on-going project. The objective of this project is to develop a vision-based inspection tool using stereo vision and AI-enabled computer vision algorithms to address the pipeline anomaly detection and characterization.

Stereo vision uses two or more cameras to extract three-dimensional (3D) information by estimating the relative depth of points observed in digital images. The principle of stereo vision is illustrated in Fig. 1. In Fig. 1(a), C1 and C2 represent the optical centers of two cameras; *b* is the baseline distance between two cameras; P is the object point; and P1 and P2 are the projection of point P in the image plane. Points C1, C2, and P form a plane known as the epipolar plane. Fig. 1(b) shows a top view of the epipolar plane where *f* is the focal length. Based on similar triangles, we have:

$$\frac{z}{f}=\frac{x}{x_l} \quad \frac{z}{f}=\frac{x-b}{x_r} \quad \frac{z}{f}=\frac{y}{y_l}=\frac{y}{y_r} \tag{1}$$

where (x,y,z) is the global coordinate of the object point P, and (x_l,y_l,z_l) and (x_r,y_r,z_r) is the coordinate of the projection of point P in the left and right image planes, respectively.



(a) (b)

Fig. 1 Principle of stereo vision: (a) epipolar plane; (b) triangulation[1]

Based on the relationships given in Eq. (1), the global coordinate of point P can be calculated as:

$$z = \frac{fb}{(x_l - x_r)} = \frac{fb}{d} \quad x = \frac{b}{d} x_l \quad y = \frac{b}{d} y_l \tag{2}$$

where the difference $d = (x_l - x_r)$ is known as the disparity. Using Eq. (2), we can determine the depth of any scene point and thus construct a depth map of the observed scene. This method of determining depth from disparity is called triangulation. In practice, triangulation will be used to find the 3D locations of critical points on pipeline defects, from which we can estimate the distances between critical points to accurately characterize pipeline defects.

AI-enabled methods for threat detection can serve a key role in risk assessment of structures. Multi-modal analysis of a scene gives us multiple sources of information from which we can determine pertinent risks and threats. Fusing these sources could potentially give us enhanced assessments of system condition. In this report, we discuss a method to fuse multi-modal information sources for scene segmentation. Data obtained from multiple sensors can be processed, combined and manipulated in innovative ways to provide better predictions. The most commonly used definition of data fusion was proposed by the Joint Directors of Laboratories (JDL) workshop[2]: "A multi-level process dealing with the association, correlation, combination of data and information from single and multiple sources to achieve refined position, identify estimates and complete and timely assessments of situations, threats and their significance." This is relevant for our project as we use depth and color information jointly for making predictions on pipeline condition. Currently, we implement a simple fusion technique for both data sources after minimal processing. However, future work would focus more on improving the ways in which we combine data sources. The method explored in this report is unsupervised, and parameter free, which yields a fully automatic analysis of the input scene.

The objective of the research in the 2nd quarter is to: (1) develop an algorithm to estimate distance using stereo vision; (2) conduct experiment to verify the effectiveness and accuracy of the developed algorithm; (3) propose an algorithm to perform unsupervised segmentation of scenes using color and disparity information;(4) evaluate the proposed image segmentation algorithm on a benchmark dataset.

2. Task 1: Development of A Novel Multi-Camera Stereo Vision System for Pipeline Inline Inspection

2.1 Algorithm for distance estimation using stereo vision

A camera projects 3D points in real world to 2D points on image plane by using the following mapping:

$$\boldsymbol{u} = \boldsymbol{K}[\boldsymbol{R}|\boldsymbol{T}]\boldsymbol{X} \tag{3}$$

where $\boldsymbol{K}$ is the 3-by-3 camera intrinsic matrix comprised of intrinsic parameters including the focal length and principle point; $\boldsymbol{R}$ and $\boldsymbol{T}$ are the 3-by-3rotation matrix and 3-by-1 translation vector (extrinsic parameters), respectively; the vector $\boldsymbol{u} = [x \quad y \quad 1]$ represents the 2D coordinates of the points in the image plane; $\boldsymbol{X} = [X \quad Y \quad Z \quad 1]$ represents the 3D coordinates of the points in the real world. Eq. (3) can be simplified as:

$$\boldsymbol{u} = \boldsymbol{P}\boldsymbol{X} \tag{4}$$

where $\boldsymbol{P}$ is the 3x4 camera projection matrix. In a stereo vision system, an object point in real world is mapped onto image planes by the two cameras as:

$$\boldsymbol{u} = \boldsymbol{P_1}\boldsymbol{X} \tag{5}$$

$$\boldsymbol{v} = \boldsymbol{P_2}\boldsymbol{X} \tag{6}$$

where $\boldsymbol{X}$ is the 3D coordinate of the object point; $\boldsymbol{P_1}$ and $\boldsymbol{P_2}$ are the camera projection matrices of the two

cameras, respectively; and $\boldsymbol{u}$ and $\boldsymbol{v}$ are the 2D coordinates of the object point in the image plane of cameras 1 and 2, respectively. Recall that the two vectors in the same direction have a cross product of zero. Thus, we have:

$$\boldsymbol{u} \times \boldsymbol{P_1 X} = \boldsymbol{0} \tag{7}$$

$$\boldsymbol{v} \times \boldsymbol{P_2 X} = \boldsymbol{0} \tag{8}$$

which can be expressed as:

$$\boldsymbol{u} \times \boldsymbol{P_1 X} = \begin{bmatrix} y_u \boldsymbol{p_1^{3T}} - \boldsymbol{p_1^{2T}} \\ \boldsymbol{p_1^{1T}} - x_u \boldsymbol{p_1^{3T}} \\ x_u \boldsymbol{p_1^{2T}} - y_u \boldsymbol{p_1^{1T}} \end{bmatrix} \boldsymbol{X} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{9}$$

$$\boldsymbol{v} \times \boldsymbol{P_1 X} = \begin{bmatrix} y_v \boldsymbol{p_2^{3T}} - \boldsymbol{p_2^{2T}} \\ \boldsymbol{p_2^{1T}} - x_v \boldsymbol{p_2^{3T}} \\ x_v \boldsymbol{p_2^{2T}} - y_v \boldsymbol{p_2^{1T}} \end{bmatrix} \boldsymbol{X} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{10}$$

where $\boldsymbol{p_1^{nT}}$ and $\boldsymbol{p_1^{nT}}$ are the transpose of the nth row of the camera projection matrix $\boldsymbol{P_1}$ and $\boldsymbol{P_2}$, respectively. Combining Eq. (9) and Eq. (10) gives:

$$\begin{bmatrix} y_u \boldsymbol{p_1^{3T}} - \boldsymbol{p_1^{2T}} \\ \boldsymbol{p_1^{1T}} - x_u \boldsymbol{p_1^{3T}} \\ y_v \boldsymbol{p_2^{3T}} - \boldsymbol{p_2^{2T}} \\ \boldsymbol{p_2^{1T}} - x_v \boldsymbol{p_2^{3T}} \end{bmatrix} \boldsymbol{X} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{11}$$

Eq. (11) represents a homogeneous linear system which can be solved using Singular Value Decomposition (SVD) to obtain the 3D coordinate $\boldsymbol{X}$. Once the coordinates of two critical points $\boldsymbol{X_1}$ and $\boldsymbol{X_2}$ are known, we can then compute the distance between the two points as:

$$dist = \sqrt{(X_1 - X_2)^2 + (Y_1 - Y_2)^2 + (Z_1 - Z_2)^2} \tag{12}$$

To implement the algorithm for distance estimation, the Matlab computer vision toolbox was adopted and the code for the algorithm is given below:

```
clear all
% loading camera projection matrices of the two cameras obtained from calibration
load sterero_Cam_par

% loading stereo image pair
Img_L = 'Test_L.png';
Img_R = 'Test_R.png';
I1 = imread(Img_L);
I2 = imread(Img_R);

% undistort the image
I1 = undistortImage(I1,stereoParams.CameraParameters1);
I2 = undistortImage(I2,stereoParams.CameraParameters2);

% image rectification
[I1,I2] = rectifyStereoImages(I1,I2,stereoParams);

% input the 2D coordinate of point 1 in the left and right image (pixel coordinates)
p1_L = [];
p1_R = [];
point3d_p1 = triangulateOnePoint(p1_L, p1_R, P1, P2);
```

```
% input the 2D coordinate of point 2 in the left and right image (pixel coordinates)
p2_L = [];
p2_R = [];
point3d_p2 = triangulateOnePoint(p2_L, p2_R, P1, P2);

% compute the distance between point 1 and point 2 in mm
distanceInCM = norm(point3d_p2-point3d_p1)


function point3d = triangulateOnePoint(point_L, point_R, P1, P2)
% formulate the homogeneous linear system shown in Eq. (11)
A = zeros(4, 4);
A(1:2,:) = point_L * P1(3,:) - P1(1:2,:);
A(3:4,:) = point_R * P2(3,:) - P2(1:2,:);

% solving the system using SVD
[~,~,V] = svd(A);
X = V(:, end);
X = X/X(end);
point3d = X(1:3)';
```

2.2 Demonstration

Experiments were designed to verify the developed algorithm for distance estimation in pipeline environment. However, the access to the lab was limited due to the situation of coronavirus outbreak. Thus, experiments were conducted using available tools at home.

2.2.1 Experiment setup

For the experiment, a dual-lens USB webcam shown in Fig. 2 is adopted. Each camera has a resolution of 640x480. To obtain the camera projection matrix, multi-plane calibration is conducted. The calibration image-set include 15 pairs of images of a chessboard placed at different angles and distance from the camera as shown in Fig. 3. The calibration is done using Zhang's Camera Calibration Algorithm[3]. The reprojection errors of the calibration are plotted in Fig. 4. It can be seen that the reprojection errors are small with an average error of approximately 0.15, indicating that accurate calibration was performed. A visualization of the calibration is shown in Fig. 5. The calibrated camera parameters including the camera projection matrix, relative orientation between cameras, and rectification rotation matrix were stored for the later use in distance estimation.

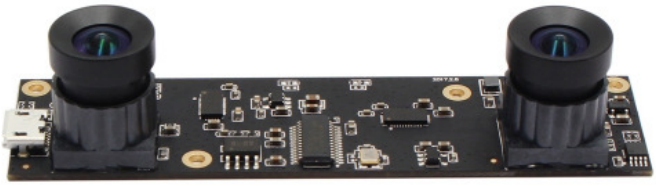


Fig. 2 Dual lens USB camera used for experiment[4]

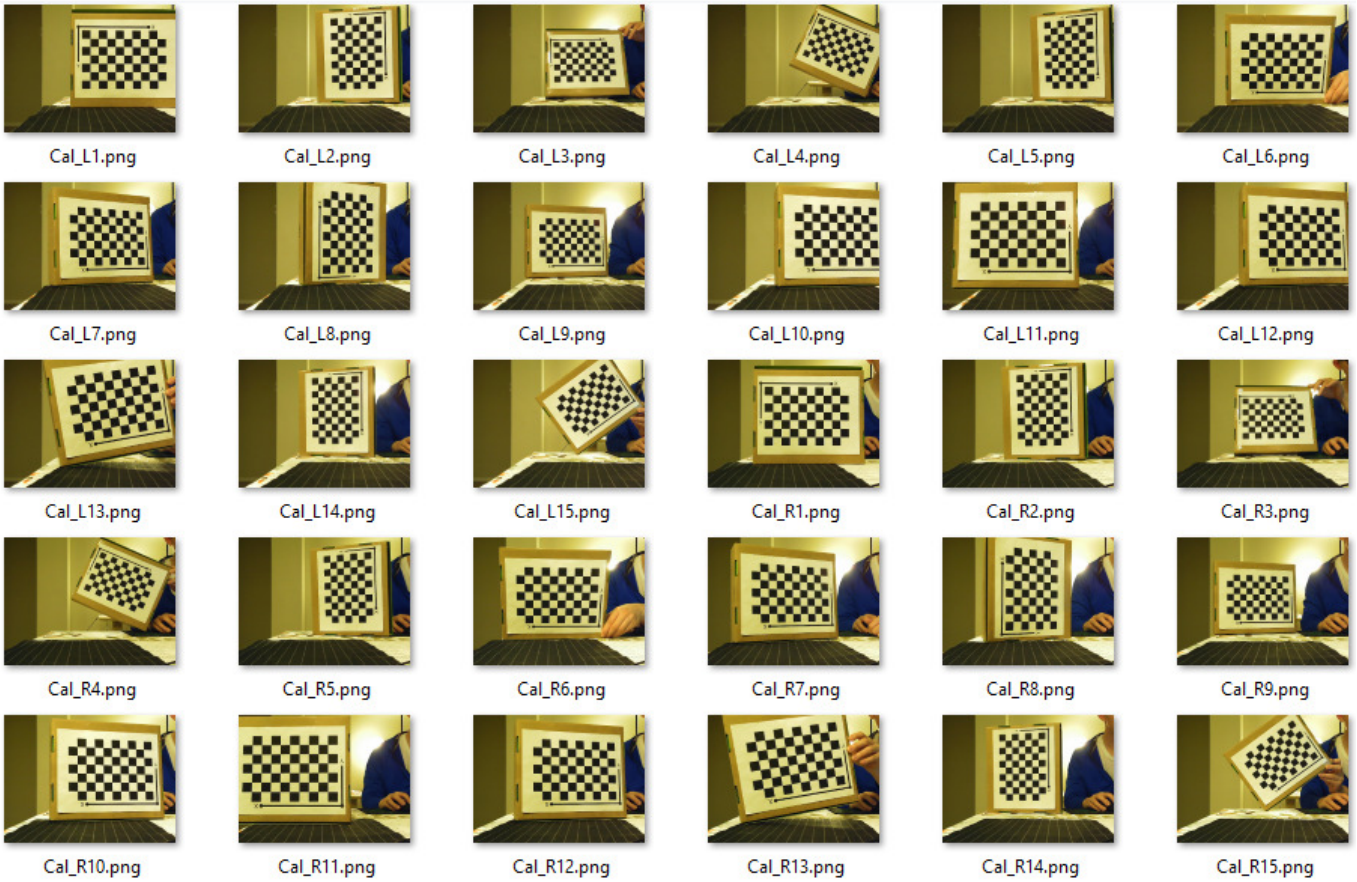


Fig. 3 Calibration images

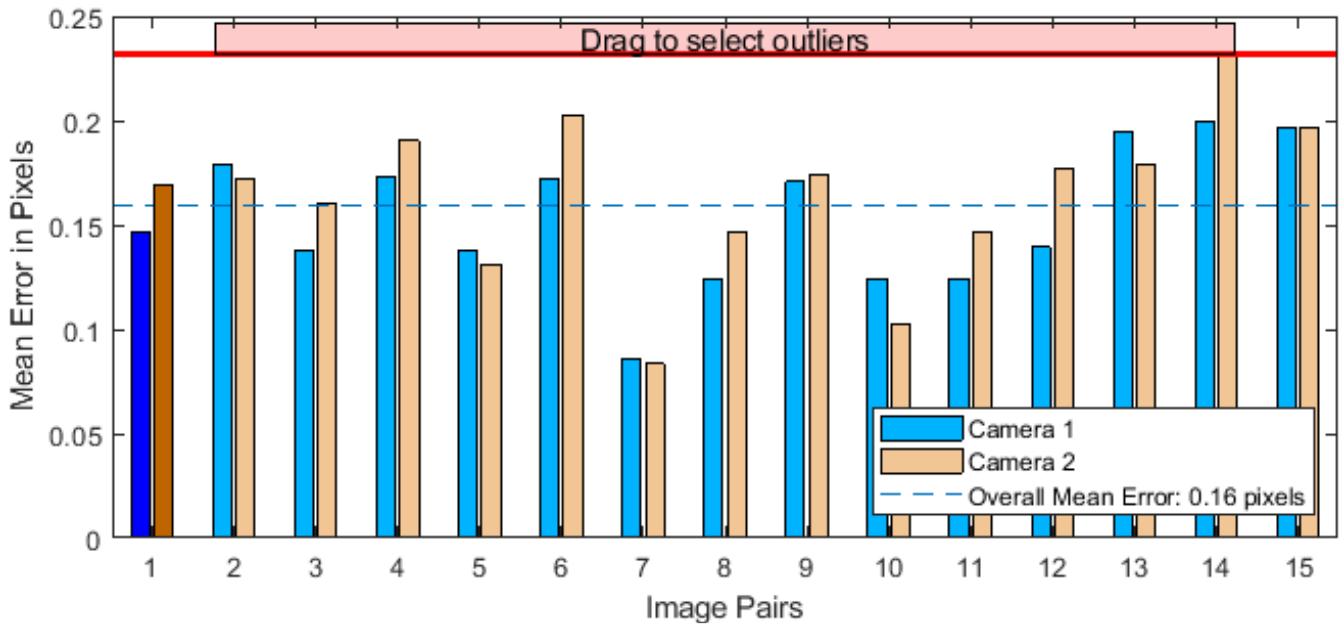


Fig. 4 Reprojection errors of calibration

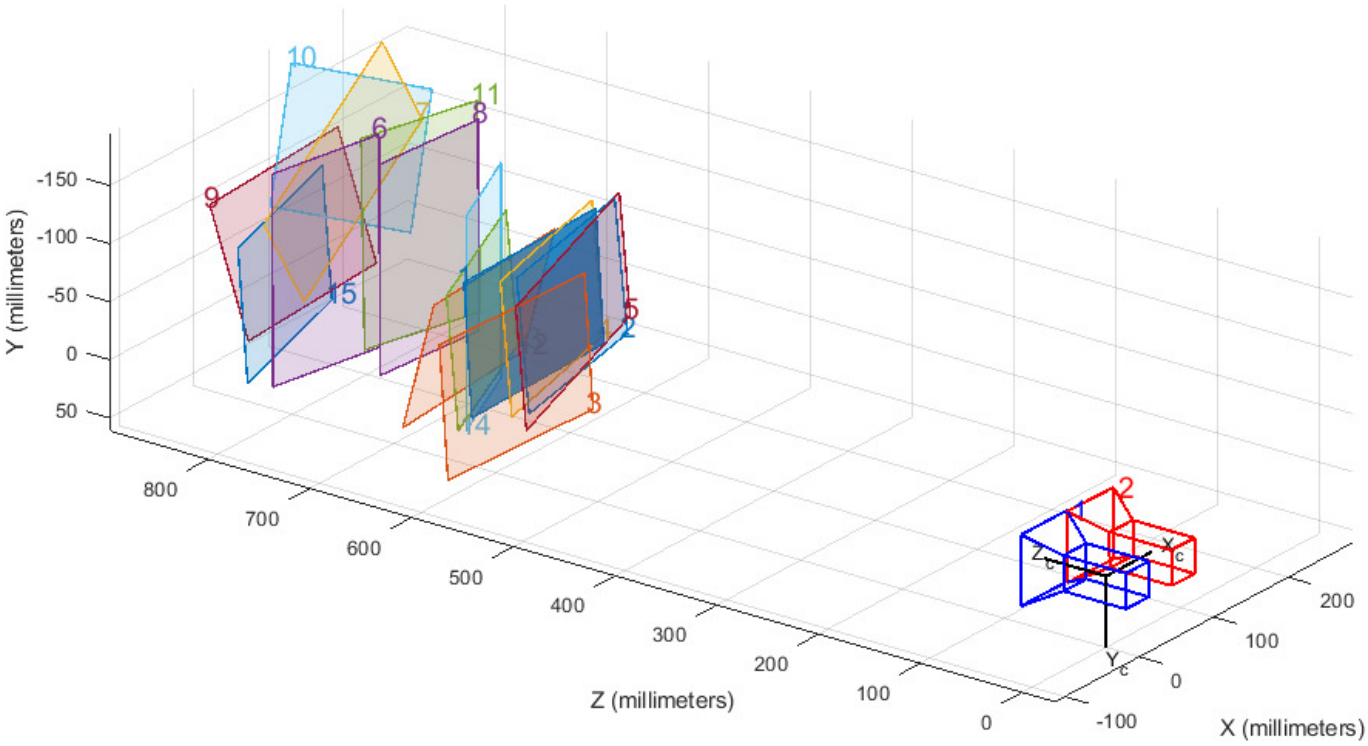
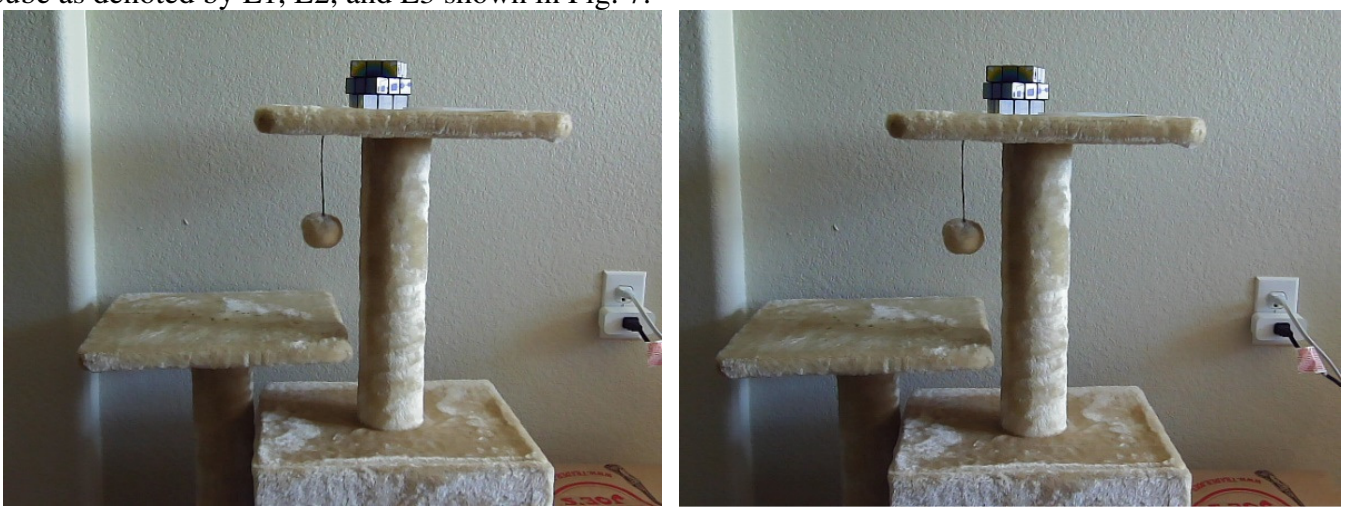


Fig. 5 Visualization of multi-plane camera calibration

The purpose of the experiment is to verify that the developed algorithm can be used to characterize the pipeline defect in terms of its size and depth. For this purpose, two case studies were designed including one case for size estimation and the other case for depth estimation. A Rubik's cube was adopted to represent the 'defect' in this experiment.

2.2.2 Case study 1

For Case study 1, the Rubik's cube is placed on a shelf as shown in Fig. 6. The top two layers of the cube are rotated to form two inclined surfaces. The objective is to estimate the edge length of each layer of the cube as denoted by L1, L2, and L3 shown in Fig. 7.



(a) (b)

Fig. 6 Image pair captured by the stereo camera for Case study 1: (a) left image; (b) right image

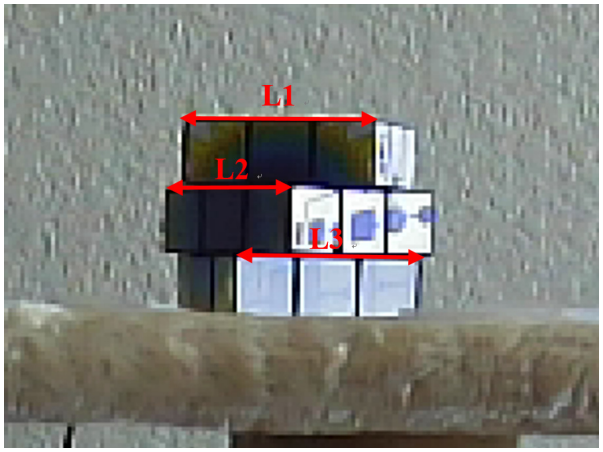


Fig. 7 Edge length of the cube

The captured image pair is rectified, and the disparity map is obtained using semi-global matching algorithm as shown in Fig. 8. From, Fig. 8, the outline of the shelf and cube can be seen from the disparity map.

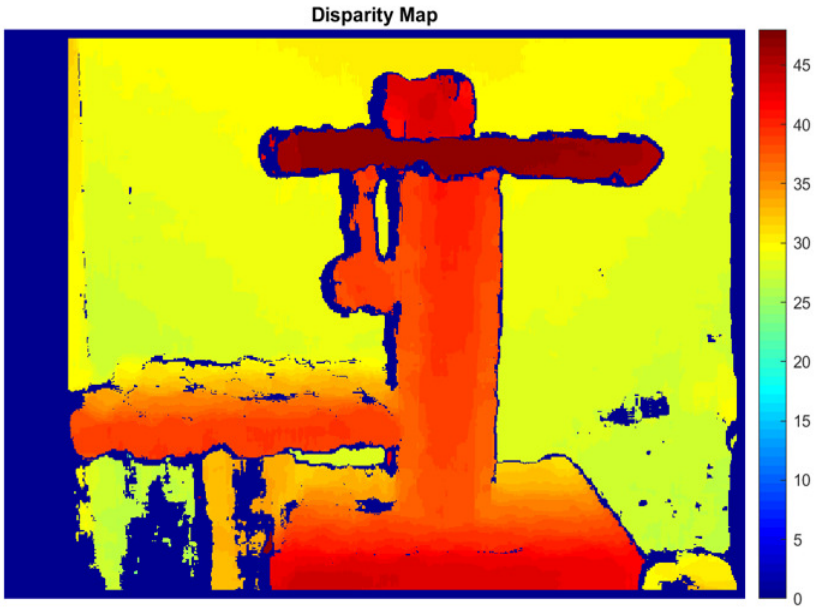


Fig. 8 Disparity map for Case study 1

The procedures for distance estimation are illustrated here by using L1 as an example. First, the 2D coordinates of the vertices are obtained from the rectified images as shown in Fig. 9. It can be seen that: (1) the 2D pixel coordinates of the left vertex of L1 are [331 42] and [288 42] in the left and right images, respectively; (2) the 2D pixel coordinates of the right vertex of L1 are [376 42] and [333 42] in the left and right images, respectively. Then, the 3D coordinates of the vertices are computed using the algorithm developed in Section 2.1. In this case, the 3D coordinates of the left and right vertices of L1 are estimated to be (21.75,-278.11, 1129.49) mm and (80.01,-277.14,1125.21) mm, respectively. Finally, the edge length is computed using Eq. (12). The edge length L1 is estimated to be 5.84 cm while the true length is measured at 5.62 cm.

(a) (b)

Fig. 9 Location and 2D coordinates of the vertices of L1: (a) rectified left image; (b) rectified right image

The same procedures are repeated for all three edges shown in Fig. 7 and the results are summarized in Table 1. It can be seen that the developed algorithm can estimate the edge length with good accuracy. The error for L2 is relatively larger due to the higher degree of inclination.

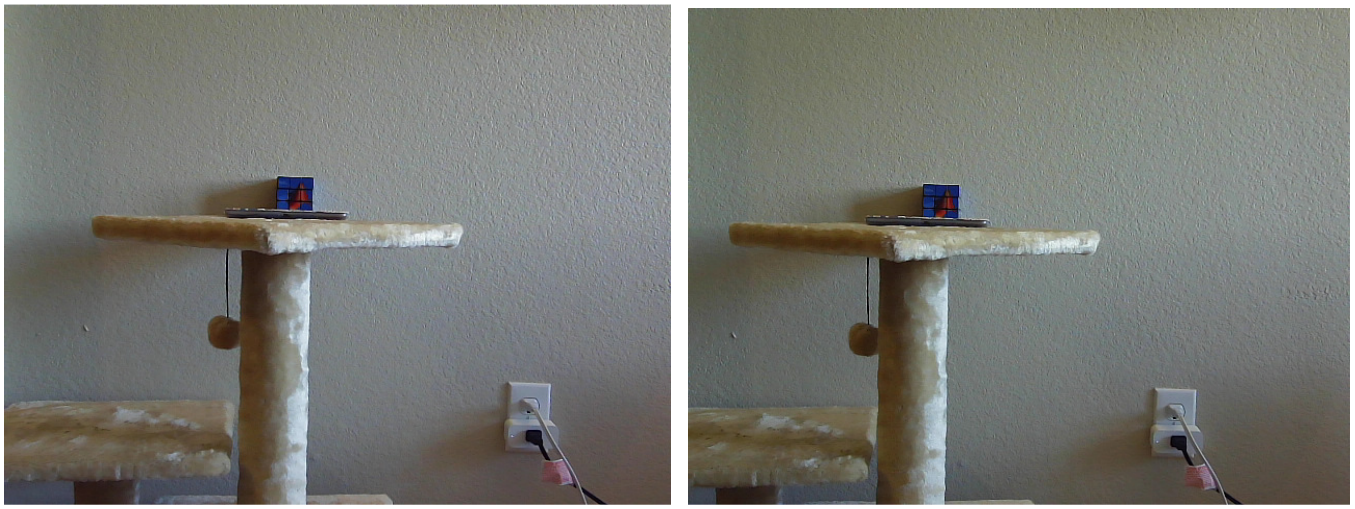
Table 1. Distance estimation results

Edge	Left vertex coordinate (mm)	Right vertex coordinate (mm)	Distance (cm)	Error (%)
L1	(21.75, -278.11, 1129.49)	(80.01,-277.14,1125.21)	5.84	3.97
L2	(16.91,-244.43,1154.33)	(51.72,-233.52,1102.62)	6.33	12.61
L3	(37.30,-236.19,1127.57)	(94.17,-235.51,1123.99)	5.70	1.40

The results of this case study demonstrate the potential of the developed algorithm to estimate the pipeline defect size. In practice, the 3D coordinates of the edge points of the defect can be obtained using the developed algorithm and then the defect size can be computed as the distance between the edge points. It should be noted that automatic detection of the pipeline defect and its corresponding edge points will be enabled by the AI-based algorithm to be developed in Task 2.

2.2.2 Case study 2

The purpose of Case study 2 is to demonstrate that the developed algorithm can estimate the depth of 'defect'. For this purpose, the Rubik's cube is placed against wall to simulate a defect protruding out of the pipeline surface as shown in Fig. 10. In this case, the depth of the 'defect' to be estimated is equal to the edge length of the cube.



(a) (b)

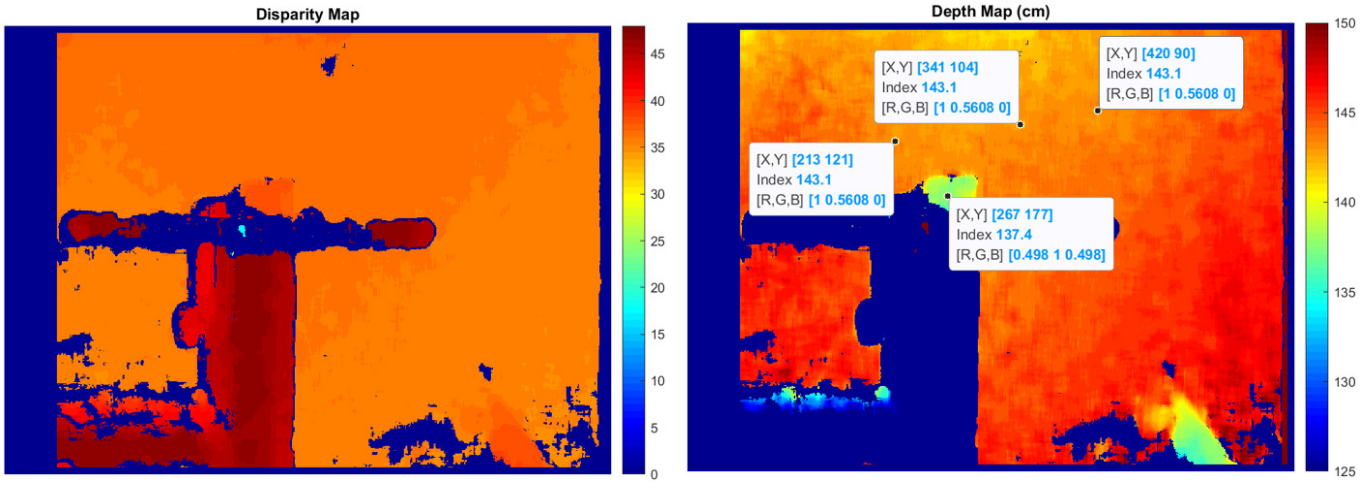
Fig. 10 Image pair captured by the stereo camera for Case Study 2: (a) left image; (b) right image

For depth estimation, the disparity map is first obtained and shown in Fig. 11 (a). Based on the rule of triangulation, the depth is computed as:

$$z = \frac{fb}{d} \tag{13}$$

where d is the disparity; f is the focal length; and b is the baseline distance between the two cameras. From the camera calibration, the focal length is found to be 869.31 pixels and the baseline distance between the two cameras is measured at 59.77 mm. Therefore, the depth of all pixels can be obtained using Eq. (13) and the depth map are plotted in Fig. 11 (b).

The depth of several representative points is denoted in Fig. 11 (b) and it can be seen the depth of wall is approximately 143.1 cm while the depth of the cube is approximately 137.4 cm, based on which the depth of the 'defect' is estimated to be 5.7 cm which is very close to the true value of 5.62 cm. It should be noted that the estimated depth are approximate values because the cameras plane is aligned approximately parallel to the wall only by hand. In practice, more rigorous alignment will be performed to ensure that the obtained depth reflects the shortest distance from the camera plane to the defect surface.



(a) (b)

Fig. 11 'Defect' Depth estimation: (a) disparity map; (b) depth map

3. Task 2: AI-enabled Anomaly Detection and Physics-based Damage Prognostics for Pipeline Integrity Management

3.1 Algorithm for unsupervised image segmentation

Traditional scene segmentation problems were tackled using only one source of data, the color image. The ability to integrate many sources of information, however, lends itself useful to this problem. We use an additional information source, the disparity map, to aid the segmentation algorithm to produce better segmentations in each frame. Fusing multiple sources of data helps us obtain a more complete picture of all the variables involved in a problem. In the case of threat analysis in pipelines using vision-based inspection techniques, depth information can potentially provide a new layer of information which can be leveraged for unique insights in the condition of pipeline systems. Stereovision algorithms enable us to use multi-camera setup to obtain multiple images, that can then be used to compute a disparity map between these images. We can use the disparity maps to create a depth map of the objects in the scene. In this section, we review the ways in which this additional per-pixel depth data can be used to enhance the data obtained from the 2-D images.

In this method, we use the combination of disparity and color information to produce segmentations. The method is parameter-free, unsupervised, and scalable to large images. Supervised techniques for segmentation, especially in complex scenes, is optimal when we used Deep Learning methods. However, training these deep models requires thousands of training examples, and has millions of parameters. Supervised models can be employed once enough problem-specific data is available, as these models produce superior detection accuracy. In case we do not have enough data, resorting to finding underlying structure in the data becomes the next best option for analysis. Unsupervised learning aims to find groupings of similar data in a dataset. This is very subjective, as the notion of similarity is problem-specific, data-specific and program-dependent. Unsupervised techniques optimize a dataset according to a user-specified condition, usually a similarity criterion, to give us clusters of similar subsets in the data. We apply this principle to the Middlebury dataset, a standard benchmark for depth estimation from color images. The dataset consists of 21 images along with a ground-truth disparity map set. All images are from indoor scenes of common objects. The scenes have complex textures, clutter and multiple objects of interest. The pipeline data would be relatively simpler in comparison, and here, we develop a general approach to dealing with segmenting images.

The flowchart of the proposed algorithm is shown in Fig. 12. The algorithm takes as input an RGB image, and a disparity map. The data sources are normalized and fused into a joint representation. This representation is transformed into an R^{NxD} matrix, which is then clustered according to [5] .The obtained segmentation is assessed qualitatively, and a metric is assigned to it. This metric is used to adjust a weighting parameter for the disparity map, and we arrive automatically at the recommended weight and best segmentation for that weight.

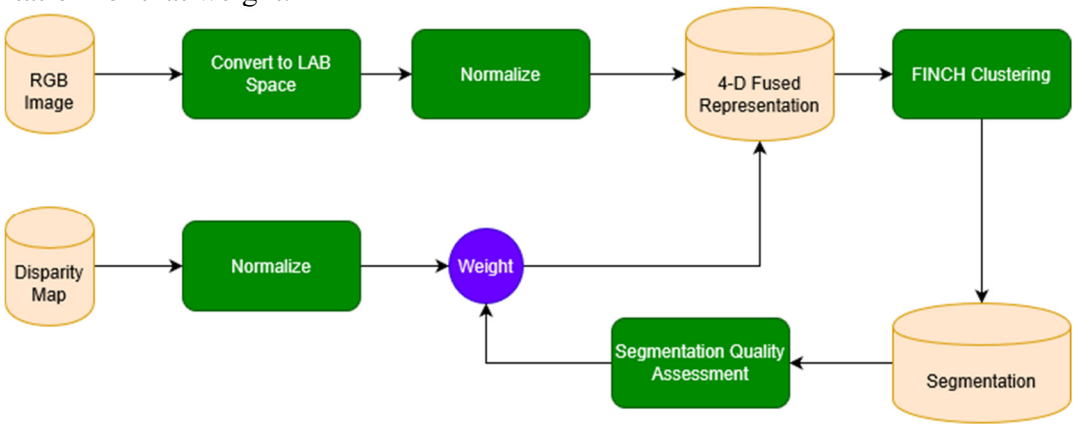


Fig. 12 Algorithm flowchart for unsupervised image segmentation using a fused representation of color and disparity data

The color image is converted first to the Lab color space from the BGR color space. The Lab color space represents the image in a form amenable to segmentation of natural scenes. While the BGR color space contains 3-color channels, Blue, Green and Red, the Lab color space divides itself into – Lightness, and 2-color channels. The lightness channel is equivalent to the intensity of the color, and the a-channel represents the color range from green to magenta. The b-channel represents the color range from blue to yellow. This way, a scene with the sky has a negative value of b-channel and a neutral value of a-channel. Scenery such as forests are easily segmented too, because the value of the a-channel is negative and the b-channel is positive. We can therefore make predictions and clusters on the Lab color space much more effectively compared to the BGR color space when it comes to natural images. In case of pipeline data, however, the use of the HSV color space often supersedes the Lab color space and the BGR color space, as we have found that corrosion detection is amenable to varying Hue and Saturation.

The input image and the disparity map are both normalized to [0,1] by using the following transformation:

$$I_{norm}(i,j) = \frac{I(i,j) - \max(I)}{\max(I) - \min(I)} \tag{14}$$

Normalization helps in maintaining standardization of value ranges across different data sources, and clustering becomes easier. The normalized data needs to then be jointly represented in some form for the clustering algorithm to operate on it. The simplest way to join the two sources is concatenation. While concatenating the two sources, we weight the disparity map with a scalar multiplier λ, which would help us adjust the relative importance of disparity with respect to color.

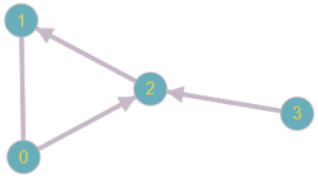
$$F = \begin{bmatrix} L \\ a \\ b \\ \lambda D \end{bmatrix} \tag{15}$$

The fused representation is recast from R^{KxLx4} into $R^{(KxL)x4}$, where K and L are the width and length of the image respectively. Let $N = KxL$, the number of samples to be clustered. We then have a 4-attribute data matrix. Clustering is an unsupervised technique to discover underlying structure in the data. Center-based methods such as K-means, and bottom-up merging techniques such as hierarchical agglomerative clustering have been used extensively for general clustering operations. K-means generally requires the number of clusters to be prespecified, but one can get around it by using information theoretic criteria to automatically estimate the number of clusters. K-means is essentially an optimization problem that minimizes within-cluster variance in an iterative fashion, by computing cluster centroids and combining the closest points. Hierarchical agglomerative clustering[6] is a bottom-up, hierarchical clustering technique that requires a distance function and a distance threshold for merging. Distance matrices on the hierarchical clustering methods are dense and scale up very quickly. Hierarchical agglomeration scales as $O(N^2 logN)$, which makes analyzing large datasets very memory and compute intensive. Sarfraz et. al.[5] came up with a new method to cluster datasets that is fast, scalable and does not require any parameter. This method belongs in the family of hierarchical clustering, as it gives us a family of clustering partitions, each with varying levels of granularity. In this method, we use the final solution as our desired output. The parameter-free method is based on discovering semantic relations between data points. The main claim that the authors make is that the first neighbor of each point is a sufficient statistic to discover structure in the data. The remainder of the algorithm takes advantage of this hypothesis in an elegant fashion to lend the entire clustering method stable, scalable and fast. We first compute the distances between all the data points, and then we find the closest neighbor to each data point. Given the index of the closest neighbor to each data point, we can define an adjacency matrix as follows:

$$A(i,j) = \begin{cases} 1, & j = \kappa_i^1 \mid i = \kappa_j^1 \mid \kappa_i^1 = \kappa_j^1 \\ 0, & otherwise \end{cases} \tag{16}$$

κ_i^1 - first neighbor of point i
κ_j^1 - first neighbor of point j

The adjacency matrix essentially creates a 1-nearest neighbor and shared neighbor relationship for the entire dataset as shown in Fig. 13. The adjacency matrix is also sparse, which enables us to use the many sparse matrix computational tools at our disposal to solve large systems efficiently. The connected components can be obtained from this matrix by constructing a graph $G = (V, E)$ where the nodes are the datapoints, and the edges link the nearest or shared neighbors. Once we have the first partition, we recursively apply the adjacency equation on the obtained partition until we reach the point where we have only one cluster. If this happens, the preceding cluster family is taken to be the final partition.



$$A = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Fig. 13 Adjacency matrix (right) and the corresponding graph representation (left)

Algorithm 1 Proposed Algorithm

Input: Sample set $S = \{1, 2, \cdots, N\}$, $S \in \mathbb{R}^{N \times d}$, where N is total number of samples and each sample point is represented by d attributes or feature dimensions.

Output: Set of Partitions $\mathcal{L} = \{\Gamma_1, \Gamma_2, \cdots, \Gamma_L\}$ where each partition $\Gamma_i = \{C_1, C_2, \cdots, C_{\Gamma_i} | C_{\Gamma_i} > C_{\Gamma_i+1} \forall i \in \mathcal{L}\}$ is a valid clustering of S.

The FINCH Algorithm:

Compute first neighbors integer vector $\kappa^1 \in \mathbb{R}^{N \times 1}$ via exact distance or fast approximate nearest neighbor methods.

Given κ^1 get first partition Γ_1 with C_{Γ_1} clusters via Equation 1. C_{Γ_1} is the total number of clusters in partition Γ_1.

while there are at least two clusters in Γ_i **do**

  Given input data S and its partition Γ_i, compute cluster means (average of all data vectors in that cluster). Prepare new data matrix $M = \{1, 2, \cdots, C_{\Gamma_i}\}$, where $\mathbb{M}^{C_{\Gamma_i} \times d}$.

  Compute first neighbors integer vector $\kappa^1 \in \mathbb{R}^{C_{\Gamma_i} \times 1}$ of points in M.

  Given κ^1 get partition Γ_M of Γ_i via Equation 1, where $\Gamma_M \supseteq \Gamma_i$

  if Γ_M has one cluster **then**

    break

  else

    Update cluster labels in $\Gamma_i : \Gamma_M \rightarrow \Gamma_i$

  end if

end while

Fig. 14 Algorithm describing the FINCH algorithm[7]

The recursive step uses the data samples obtained in each cluster to compute the mean of each cluster, and the obtained mean vectors are used again to compute the first neighbor. As the only criterion used in this algorithm is the first neighbor, and the clustering algorithm is hierarchical, the algorithm is termed as the First Integer Neighbor Clustering Hierarchy (FINCH)[7]. The pseudo code of the FINCH algorithm is given in Fig. 14. The key advantage of this algorithm is that the number of clusters reduce quickly, and the computation of only one neighbor makes the algorithm run at $O(NlogN)$.

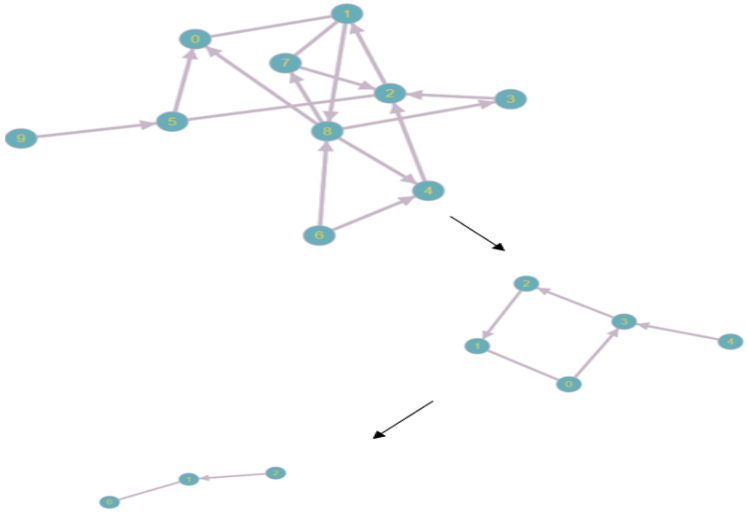


Fig. 15 Recursive cluster combining in the FINCH algorithm

The resultant segmentation is evaluated qualitatively using metric as proposed in [7].The relative importance of the disparity map with respect to the color image is weighted using the scalar λ. Since a single optimal value does not exist, we can use the metric to automatically determine a quality segmentation based on user-defined criteria. A good segmentation should have the following properties:

- Within cluster variance must be minimized
- Between cluster variance should be maximized

The above conditions are satisfied by first normalizing the greyscale image and the disparity map to [0,1]. Let us consider the within cluster variance calculation for color. Given an image I, we can calculate the local standard deviation of each cluster as:

$$\sigma_t(S_i) = \frac{\sum_{j \epsilon S_i} \sigma_w(j)}{|S_i|} \tag{17}$$

$\sigma_t(S_i)$ is a vector of k-dimension, where k is the number of clusters. The $\sigma_w(j)$ is obtained by using a 3x3 sliding window through the entire image and the cluster mask. The cluster mask windows are then evaluated for the dominant cluster in each window. If the dominant cluster of a given window is 3, then the corresponding standard deviation value of that window is stored in the 3rd index of $\sigma_t(S_i)$. Once the entire image is evaluated this way for local standard deviations, the above equation is applied by summing each entry in $\sigma_t(S_i)$ and dividing by the cardinality of the respective cluster. This is faster than directly calculating the above equation for each cluster separately.

Once we have obtained the local standard deviation $\sigma_t(S_i)$, we calculate the global standard deviation of the entire image $\sigma(S_i)$. The idea is to compute the cluster-wise standard deviation, and for that, we use the following equation:

$$D_i^{intra} = \frac{max(\sigma(S_i) - \sigma_t(S_i), 0).|S_i|}{N} \tag{18}$$

The above equation goes to zero whenever the internal cluster standard deviation is greater than the global cluster standard deviation. It has the effect of reducing the weight of highly textured regions, and we end up valuing depth information more in these cases. It is better to use depth information in highly textured regions and color information in uniform color regions. The total intra-cluster score is computed as the sum of the intra-cluster measures for each cluster:

$$D_{color}^{intra} = \sum_i D_i^{intra} \tag{19}$$

The inter-cluster variances are computed as the differences between the cluster means for each cluster, which returns a square matrix of dimension KXK, where K is the number of clusters:

$$D_{i,j}^{inter} = \left|E(S_i) - E(S_j)\right| \tag{20}$$

$$D_{color}^{inter} = \frac{\sum_{i,j} D_{i,j}^{inter}}{K(K-1)} \tag{21}$$

The obtained measures are then averaged to give the color score:

$$Q^{color} = \frac{D_{color}^{inter} + D_{color}^{intra}}{2} \tag{22}$$

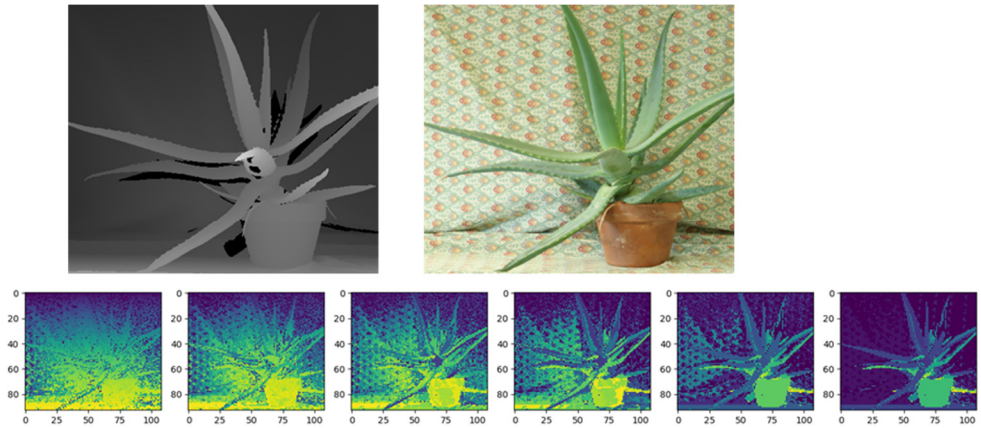
The method for obtaining geometry information is computed in a similar manner, except the intra-cluster measure calculation does not need a sliding window, but a direct standard deviation is computed for each cluster. The total quality measure is then:

$$Q = Q^{color} + Q^{disparity} \tag{23}$$

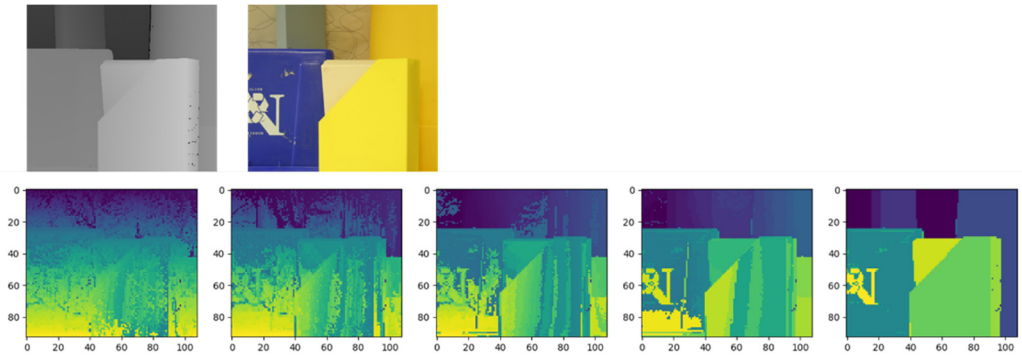
For our analyses, we use the value of λ corresponding to the highest Q value.

3.2 Results and Discussion

Fig. 16 shows the first set of results which demonstrate how the FINCH algorithm produces a family of clustering partitions recursively. Earlier clusters consist of a noisy representation of the image, with many clusters being identified, and later clusters refine themselves slowly to converge to a final clustering. The use of the first neighbor relation makes the clustering process very quick, and the use of sparse matrix operations is taken full advantage of. For instance, the final clustering consists of only 17 clusters, and the first clustering consists of 1056 clusters, in case of the Plastic box image.



(a)



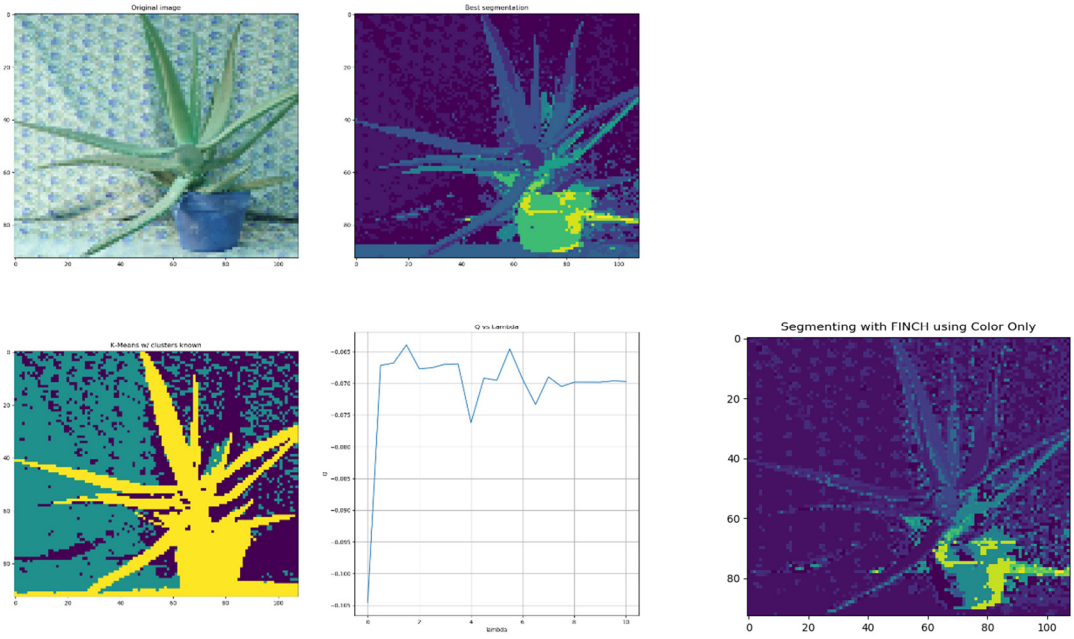
(b)

Fig. 16 Demonstration of the FINCH hierarchy in the Middlebury dataset

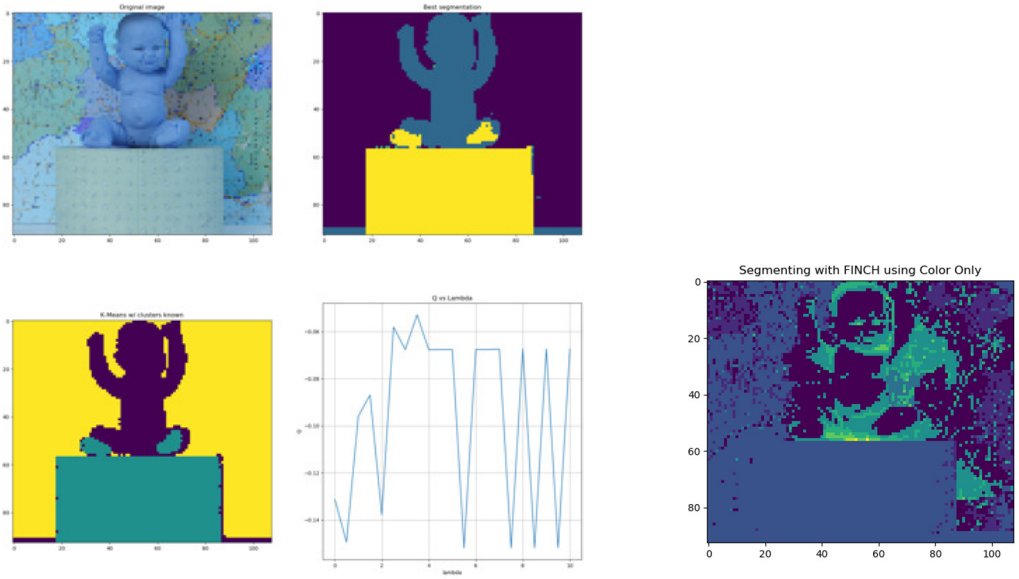
We now examine some results of the algorithm on the Middlebury dataset. Fig. 17 provides 5 examples of results from the dataset. Each example of Fig. 17 contains an image to be segmented (top left), its corresponding FINCH segmentation with the best weight for the disparity map(top right), the k-means clustering solution given the number of ground truth clusters is known (bottom left), a plot of the Q-values across λ (bottom middle), and the FINCH solution when we use only color information for clustering instead of both color and disparity information jointly. A few general observations across all images are notable. The k-means clustering algorithm given the number of clusters as equal to the ground truth produces good quality clusters on all images with both the disparity map and the color image data. The hierarchical FINCH clustering method is parameter-free and produces clusters that are at most as good as the k-means case as seen in the baby scene (Fig. 17 (b)). The other clusters are lower in quality but manage to segment out key areas of the image, as seen in Fig. 17 (a) and (d). Fig. 17 (c) and (e) however fail to meaningfully partition the image into clusters even after performing the weight selection procedure. The weight selection procedure does not produce consistently good segmentations. Modifications to the weight selection procedure are needed to ensure that this consistency is obtained. Observing the Q vs λ graph on all image demonstrations indicates that the variation in Q has no consistent pattern. Ideally, one would want a convex functional relation between Q and λ, but in this case we do not find such a relationship. This relationship needs to be corrected to obtain a good metric for weight selection. A key observation across all images is that the addition of disparity information greatly improves segmentation quality for FINCH clustering. Table 2 shows us that there is a reduction in the number of clusters when adding disparity information along with color, and well-segmented images show a greater reduction in this number as compared to poorly segmented cases.

Table 2. Comparison of ground truth cluster number, and the reduction achieved when color and disparity is combined for the FINCH algorithm

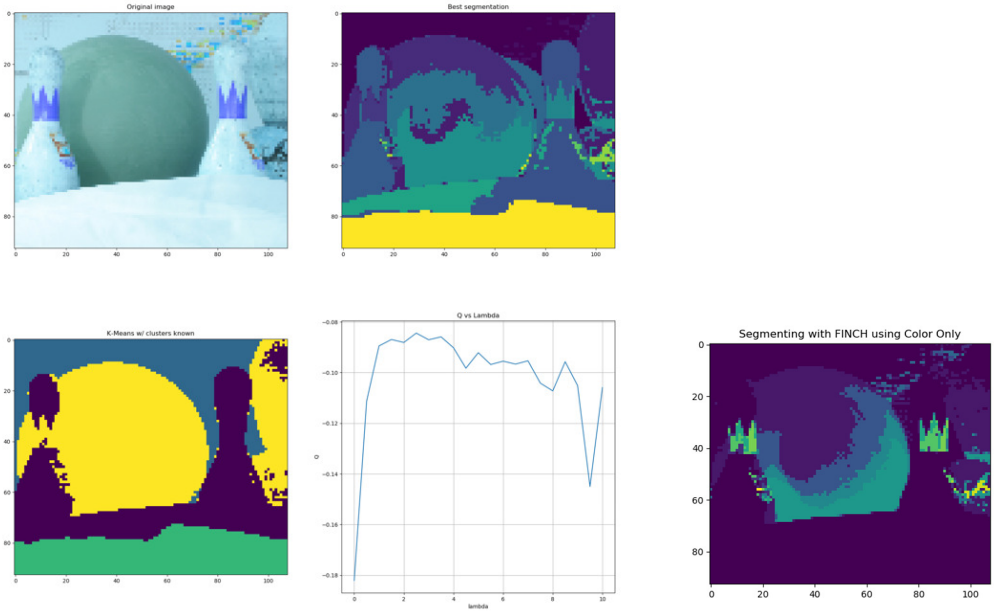
Scene Name	Ground Truth Clusters #	FINCH with only color	FINCH with both color and disparity
Aloe	3	24	11
Baby	3	9	4
Bowling	4	16	7
Plastic case	4	22	17
Flowerpot	2	18	16



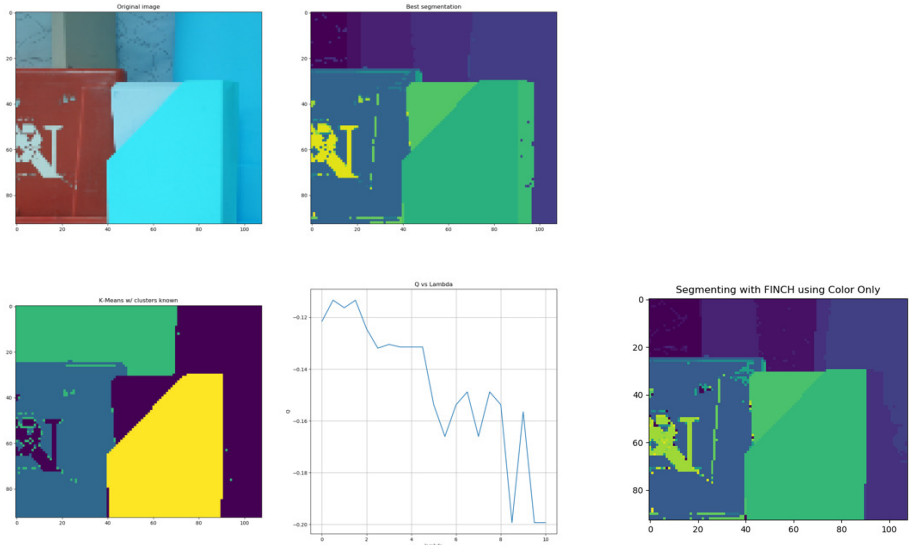
(a)


Original image
Best segmentation
K-Means w/ clusters known
Q vs Lambda
Q
lambda
Segmenting with FINCH using Color Only

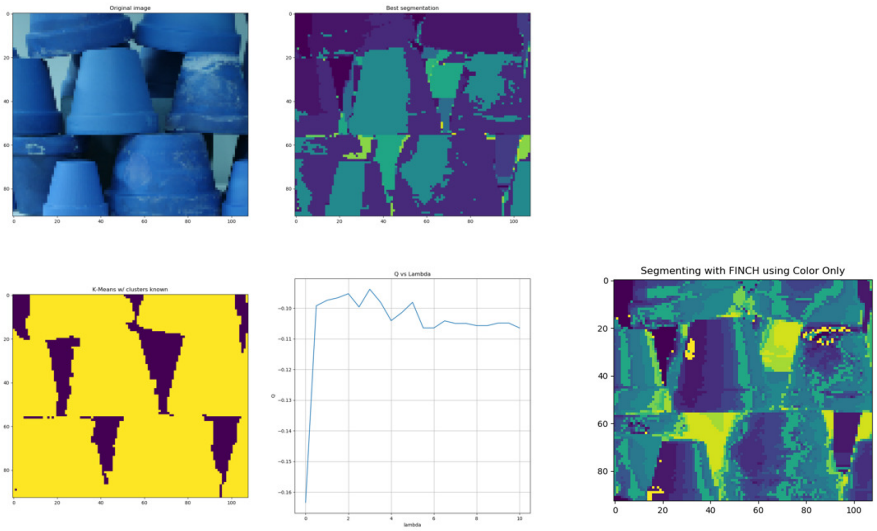
(b)


Original image
Best segmentation
K-Means w/ clusters known
Q vs Lambda
Q
lambda
Segmenting with FINCH using Color Only

(c)


Original image
Best segmentation
K-Means w/ clusters known
Q vs Lambda
Q
lambda
Segmenting with FINCH using Color Only

(d)



(e)

Fig. 17 Demonstrative examples of image segmentation using the FINCH algorithm using color and disparity information fusion on the Middlebury Dataset.

4. Summary and Future Work

4.1 Task 1

In this quarter, we developed the algorithm for estimating distance using stereo vision and demonstrated its potential in characterizing pipeline defects in terms of size and depth through experiments. In the next quarter, we will conduct experiment in pipelines to simulate realistic environment for pipeline inspection. Furthermore, we will continue to work on hardware integration and investigate hardware requirements particularly on lighting quality and camera resolution to ensure accurate detection and characterization of pipeline defects.

4.2 Task 2

In this quarter, we discussed an unsupervised algorithm that can take a color image and its corresponding disparity map and deliver a segmentation without the use of any parameters. We qualitatively evaluated its efficacy in comparison to k-means clustering and considered the effect of adding disparity information to the segmentation results. Future progress would be focused on improving segmentation capabilities by proposing a new, improved weighting metric. Another area of focus would be on data fusion techniques and looking for better ways to combine information before they are clustered. We are also going to be working with corrosion data and are considering including simulated disparity maps for existing open-source corrosion data on the internet. This would be helpful, as we can evaluate these general segmentation methods on our specific problem.

References

[1] University of Washington. Lecture 16: Stereo and 3D Vision of CSE455: Computer Vision. 2018.
[2] White FE. Data Fusion Lexicon. Data Fusion Subpanel Jt. Dir. Lab. Tech. Panel C3, 1991.
[3] Burger W. Technical Report HGB16-05: Zhang's Camera Calibration Algorithm: In-Depth Tutorial and Implementation. Hagenberg, Austria: 2016.
[4] ELP. ELP No Distortion Dual Lens USB Webcam 1920x1080 Aptina AR0330 Mini 86*23mm USB Camera Module For Biometric Retina Retina Analyze n.d. http://www.webcamerausb.com/elp-no-distortion-dual-lens-usb-webcam-1920x1080-aptina-

ar0330-mini-8623mm-usb-camera-module-for-biometric-retina-retina-analyze-p-288.html.
[5] Sarfraz S, Sharma V, Stiefelhagen R. Efficient parameter-free clustering using first neighbor relations. Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., 2019. doi:10.1109/CVPR.2019.00914.
[6] Müllner D. Fastcluster: Fast hierarchical, agglomerative clustering routines for R and Python. J Stat Softw 2013;53. doi:10.18637/jss.v053.i09.
[7] Mutto CD, Zanuttigh P, Cortelazzo GM. Fusion of Geometry and Color Information for Scene Segmentation. IEEE J Sel Top Signal Process 2012;6:505–21. doi:10.1109/JSTSP.2012.2194474.